COLLECTION FRAMEWORK :

01 March 2025 10:26

1. What is collection ?
   - It is single entity or object in which we store multiple data

2. What is Framework ?
   - It represent the library (predefined classes and interfaces).

- **Collection Framework** is a **ready-made set of classes and interfaces** provided by Java to **store, manage, and process a group of objects easily**.

It contains 2 main parts :

- Java.util.Collection -> we can store data directly.
- Java.util.Map -> we can store in the form of key value pair.

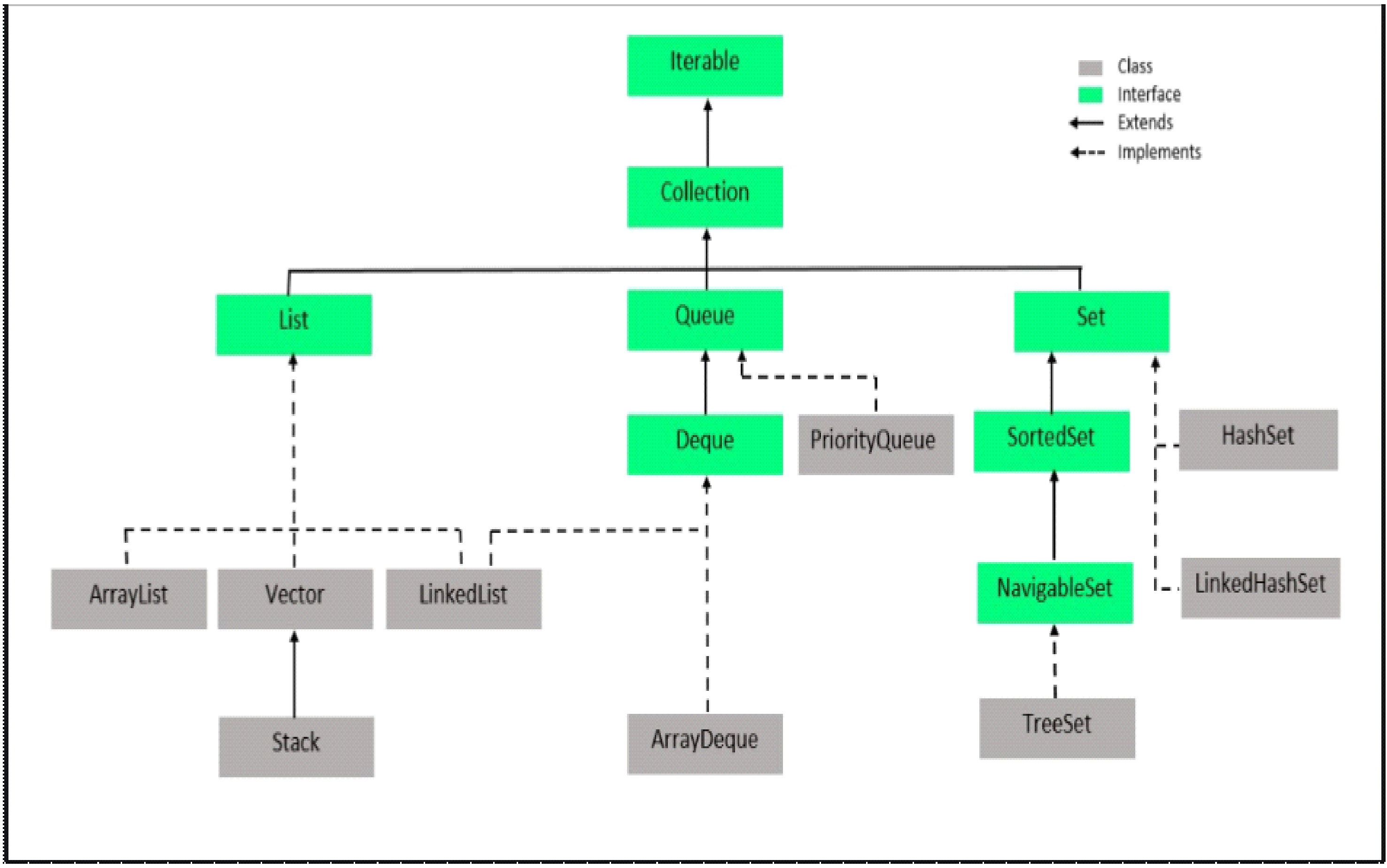
Difference between Collection and Collections

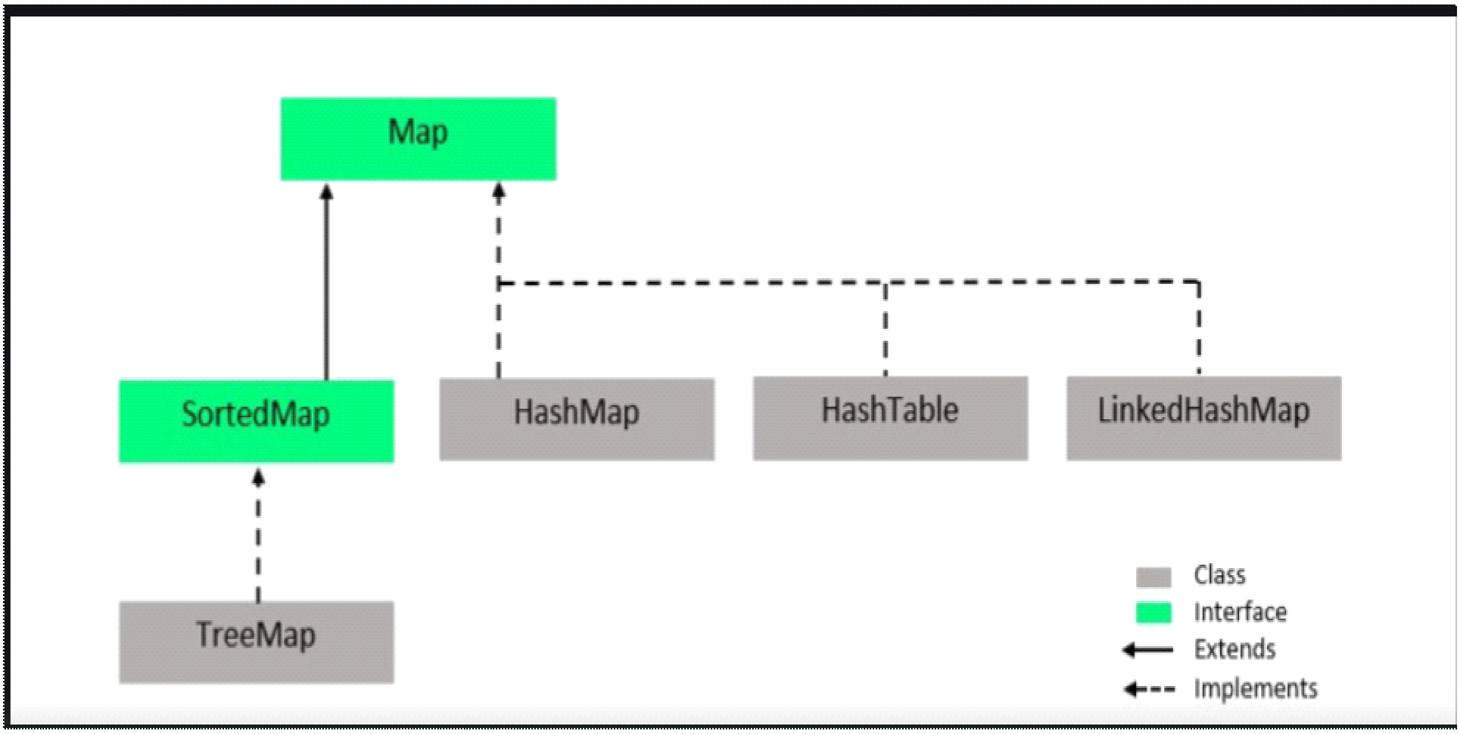
1 **Collection**

- **It is an interface**.
- It is part of **java.util package**.
- It is the **parent interface** of List, Set, and Queue.
- It defines common methods like add(), remove(), size(), etc.
- Example: List, Set, Queue all extend Collection.

2 **Collections**

- **It is a class**.
- It is part of **java.util package**.
- It is a **utility/helper class** to work with collections.
- It provides **static methods** for sorting, searching, shuffling, etc.
- Example: Collections.sort(list)





- **Collection Interface Methods (Point-wise)**
  1. **add(E e)** - Adds the given element to the collection.
  2. **addAll(Collection<? extends E> c)** - Adds all elements from the given collection to the current collection.
  3. **clear()** - Removes all elements from the collection.
  4. **contains(Object o)** - Checks if the collection contains the specified element.
  5. **containsAll(Collection<?> c)** - Checks if the collection contains all elements from the given collection.
  6. **isEmpty()** - Checks if the collection is empty (size is zero).
  7. **iterator()** - Returns an Iterator to traverse through the collection.
  8. **remove(Object o)** - Removes the specified element from the collection.
  9. **removeAll(Collection<?> c)** - Removes all elements from the current collection that are also present in the given collection.
  10. **retainAll(Collection<?> c)** - Retains only the elements in the current collection that are also present in the given collection (removes others).
  11. **size()** - Returns the total number of elements in the collection.
  12. **toArray()** - Converts the collection into a general Object array.
  13. **toArray(T[] a)** - Converts the collection into a specific type array (T type array).
  14. **stream()** - Converts the collection into a sequential Stream (available from Java 8).

CODE :

```java
package collection_framework;

import java.util.*;

public class CollectionMethodsDemo {
    public static void main(String[] args) {

        Collection<String> collection = new ArrayList<>();

        // 1. add(E e)

        collection.add("Apple");
        collection.add("Banana");
        System.out.println("After add(): " + collection);

        // 2. addAll(Collection<? extends E> c)

        Collection<String> moreFruits = Arrays.asList("Mango", "Orange");

        collection.addAll(moreFruits);
        System.out.println("After addAll(): " + collection);

        // 3. clear()

        Collection<String> temp = new ArrayList<>(collection); // backup for later use
```

```java
        collection.clear();
        System.out.println("After clear(): " + collection);

        // Restore collection for further examples

        collection.addAll(temp);

        // 4. contains(Object o)

        System.out.println("Contains 'Apple': " + collection.contains("Apple"));

        // 5. containsAll(Collection<?> c)

        System.out.println("Contains all [Apple, Mango]: " +
collection.containsAll(Arrays.asList("Apple", "Mango")));

        // 6. isEmpty()

        System.out.println("Is collection empty? " + collection.isEmpty());

        // 7. iterator()

        System.out.print("Using iterator: ");

        Iterator<String> iterator = collection.iterator();

        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }

        System.out.println();

        // 8. remove(Object o)

        collection.remove("Banana");
        System.out.println("After remove('Banana'): " + collection);

        // 9. removeAll(Collection<?> c)

        collection.removeAll(Arrays.asList("Mango", "Orange"));
        System.out.println("After removeAll(): " + collection);

        // 10. retainAll(Collection<?> c)

        System.out.println("before and before: " + collection);
        collection.addAll(Arrays.asList("Grapes", "Pineapple", "Apple"));

        System.out.println("before retainAll(): " + collection);

        collection.retainAll(Arrays.asList("Apple", "Grapes"));
        System.out.println("After retainAll(): " + collection);

        // 11. size()

        System.out.println("Collection size: " + collection.size());

        // 12. toArray()

        Object[] array = collection.toArray();
        System.out.println("toArray(): " + Arrays.toString(array));

        // 13. toArray(T[] a)

        String[] stringArray = collection.toArray(new String[0]);
        System.out.println("toArray(T[] a): " + Arrays.toString(stringArray));

//        By passing new String[0], we are saying:
//        "We don't know the exact size, so start with an empty array (size 0), and Java will create the correct
sized array for us."

        // 14. stream() - (Java 8+)
```

```java
        System.out.print("Using stream(): ");
        collection.stream().forEach(System.out::println);

    }
}
```